RECEIVED
CENTRAL FAX CENTER
AUG 13 2007

IN THE CLAIMS

Please cancel withdrawn claims 74 and 81-122 and amend claims 1, 37, 39, 75 and 80 as follows:

1. (CURRENTLY AMENDED) A method for solving a system of N linear equations in N unknown variables, the method comprising:

(a) storing an estimate value for each unknown variable;

(b) initialising each estimate value to a predetermined value;

(c) for each estimate value:

(i) determining whether a respective predetermined condition is satisfied; and

(ii) updating the estimate if and only if the respective predetermined condition is satisfied; ~~and~~

(d) repeating step (c) ~~until each estimate value is sufficiently close to an accurate value of the respective unknown variable~~ a plurality of times; and

outputting said estimate values to provide an estimate of a solution to said system of linear equations.

2. (ORIGINAL) A method according to claim 1, wherein said updating comprises adding a scalar value d to the respective estimate value, or subtracting a scalar value d from the respective estimate value.

3. (ORIGINAL) A method according to claim 2, wherein said scalar value d is updated in a predetermined manner.

4. (ORIGINAL) A method according to claim 3, wherein said scalar value d is updated when and only when step (c) updates no estimate values.

5. (ORIGINAL) A method according to claim 4, wherein said updating divides d by a scalar update value.

6. (ORIGINAL) A method according to claim 5, wherein the scalar update value is equal to a power of two.

7. (ORIGINAL) A method according to claim 6, wherein the scalar update value is equal to two.

8. (ORIGINAL) A method according to claim 1, wherein each of said estimate values is initialised to be equal to zero.

9. (ORIGINAL) A method according to claim 1, wherein the respective predetermined condition for each respective estimate value does not involve the respective estimate value.

10. (ORIGINAL) A method according to claim 2, wherein the method establishes a respective auxiliary value for each estimate value.

11. (ORIGINAL) A method according to claim 10, wherein said auxiliary values form an auxiliary vector **Q**.

12. (ORIGINAL) A method according to claim 11, wherein said predetermined condition for each respective estimate value involves the respective auxiliary value.

13. (ORIGINAL) A method according to claim 12, wherein a plurality of auxiliary values are associated with each estimate value.

14. (ORIGINAL) A method according to claim 13, wherein the predetermined condition for a respective estimate value involves the minimum amongst the plurality auxiliary values.

15. (ORIGINAL) A method according to claim 14, wherein the minimum value is compared with a threshold value.

16. (ORIGINAL) A method according to claim 15, wherein the condition is satisfied if the minimum value is less than the threshold value.

17. (ORIGINAL) A method according to claim 16, wherein the plurality of auxiliary values for a respective estimate value consist of a first auxiliary value, and second auxiliary value which is the negative of the first auxiliary value.

18. (ORIGINAL) A method according to claim 17, wherein the threshold value for the *n*th unknown variable is the scalar value *d* multiplied by the coefficient of the *n*th unknown variable in the *n*th equation.

19. (ORIGINAL) A method according to claim 18, wherein one of a plurality of updates is selected in the condition is satisfied.

20. (ORIGINAL) A method according to claim 19, wherein the scalar value *d* is added to the respective estimate value if the condition is satisfied and minimum value is the first auxiliary value.

21. (ORIGINAL) A method according to claim 19, wherein the scalar value *d* is subtracted from the respective estimate value if the condition is satisfied and minimum value is the second auxiliary value.

22. (ORIGINAL) A method according to claim 20, wherein the first auxiliary value for the *n*th unknown variable is initially set to be equal to the negative of the right hand side of the *n*th equation.

23. (ORIGINAL) A method according to claim 21, wherein the first auxiliary value for the *n*th unknown variable is initially set to be equal to the negative of the right hand side of the *n*th equation.

24. (ORIGINAL) A method according to claim 19, wherein the respective first and second auxiliary values are updated if the condition is satisfied.

25. (ORIGINAL) A method according to claim 24, wherein the first and second auxiliary values associated with each estimate value are updated if the condition is satisfied.

26. (ORIGINAL) A method according to claim 25, wherein if the predetermined condition is satisfied for the *n*th estimate value:

the first auxiliary value for the *m*th estimate value is updated by:

multiplying the coefficient of the *m*th unknown variable in the *n*th equation by the scalar value *d*; and

adding the result of said multiplication to the first auxiliary value to create a new first estimate auxiliary value, or subtracting the result of said multiplication from the first auxiliary value to create the new first estimate auxiliary value; and

the second auxiliary value for the *m*th estimate value is updated to be equal to the negative of the new first auxiliary value.

27. (ORIGINAL) A method according to claim 1, wherein each estimate value is represented as a fixed point binary word.

28. (ORIGINAL) A method according to claim 1, wherein each estimate value is a floating point binary word.

29. (ORIGINAL) A method according to claim 1, wherein each estimate value is a complex number.

30. (ORIGINAL) A method according to claim 3, wherein the scalar value *d* is updated such that the algorithm updates the estimate values in a bitwise manner, beginning with the most significant bit.

31. (ORIGINAL) A method according to claim 4, wherein step (d) is carried out until a predetermined condition is satisfied.

32. (ORIGINAL) A method according to claim 31, wherein said predetermined condition is a maximum number of iterations without an update to the scalar value *d*.

33. (ORIGINAL) A method according to claim 32, wherein said predetermined condition is a total execution time elapsed without an update to the scalar value *d*.

34. (ORIGINAL) A method according to claim 1, wherein the accurate solution of the equations is known to lie between upper and lower bounds, and the algorithm seeks a solution between said upper and lower bounds.

35. A method according to claim 34, wherein said estimate values are initialised to a value which is within said upper and lower bounds.

36. (ORIGINAL) A method according to claim 35, wherein said estimate values are initialised to a value positioned at the midpoint of said upper and lower bounds.

37. (CURRENTLY AMENDED) A computer apparatus for solving a system of *N* linear equations in *N* unknown variables, the apparatus comprising:

a program memory containing processor readable instructions; and

a processor for reading and executing the instructions contained in the program memory; wherein said processor readable instructions comprise instructions controlling the processor to:

(a) store an estimate value for each unknown variable;

(b) initialise each estimate value to a predetermined value;

(c) for each estimate value:

(i) determine whether a respective predetermined condition is satisfied; and

(ii) update the estimate if and only if the respective predetermined condition is satisfied;

(d) repeat step (c) a plurality of times; and

(e) output said estimate values to provide an estimate of a solution to said system of linear equations ~~carry out the method according to claim 1~~.

38. (ORIGINAL) A data carrier carrying computer readable program code to cause a computer to execute procedure in accordance with the method of claim 1.

39. (CURRENTLY AMENDED) A method for solving a system of *N* linear equations in *N* unknown variables, the method comprising:

(a) storing an estimate value for each unknown variable;

(b) initialising each estimate value to a predetermined value;

(c) attempting to update each estimate value using a scalar value *d*;

(d) updating the scalar value if no updates are made in step (c); ~~and~~

(e) repeating step (c) and step (d) a plurality of times; and

(f) output said estimate values to provide an estimate of a solution to said system of linear equations ~~until each estimate value is sufficiently close to an accurate value of the respective unknown variable~~.

40. (ORIGINAL) A method according to claim 39, wherein updating said estimate values comprises adding the scalar value *d* to an estimate value, or subtracting the scalar value *d* from an estimate value

41. (ORIGINAL) A method according to claim 40, wherein said updating the scalar value divides the scalar value by a scalar update value.

42. (ORIGINAL) A method according to claim 41, wherein the scalar update value is equal to a power of two.

43. (ORIGINAL) A method according to claim 42, wherein the scalar update value is equal to two.

44. (ORIGINAL) A method according to claim 39, wherein each of said estimate values is initialised to be equal to zero.

45. (ORIGINAL) A method according to claim 39, wherein step (c) comprises:
for each estimate value:

(i) determining whether a respective predetermined condition is satisfied; and

(ii) updating the estimate if and only if the respective predetermined condition is satisfied;

46. (ORIGINAL) A method according to claim 45, wherein the method establishes a respective auxiliary value for each estimate value.

47. (ORIGINAL) A method according to claim 46, wherein said auxiliary values form an auxiliary vector **Q**.

48. (ORIGINAL) A method according to claim 47, wherein said predetermined condition for each respective estimate value involves the respective auxiliary value.

49. (ORIGINAL) A method according to claim 48, wherein a plurality of auxiliary values are associated with each estimate value.

50. (ORIGINAL) A method according to claim 49, wherein the predetermined condition for a respective estimate value involves the minimum amongst the plurality auxiliary values.

51. (ORIGINAL) A method according to claim 50, wherein the minimum value is compared with a threshold value.

52. (ORIGINAL) A method according to claim 51, wherein the condition is satisfied if the minimum value is less than the threshold value.

53. (ORIGINAL) A method according to claim 52, wherein the plurality of auxiliary values for a respective estimate value consist of a first auxiliary value, and second auxiliary value which is the negative of the first auxiliary value.

54. (ORIGINAL) A method according to claim 53, wherein the threshold value for the *n*th unknown variable is the scalar value *d* multiplied by the coefficient of the *n*th unknown variable in the *n*th equation.

55. (ORIGINAL) A method according to claim 54, wherein one of a plurality of updates is selected if the condition is satisfied.

56. (ORIGINAL) A method according to claim 55, wherein the scalar value *d* is added to the respective estimate value if the condition is satisfied and minimum value is the first auxiliary value.

57. (ORIGINAL) A method according to claim 56, wherein the scalar value *d* is subtracted from the respective estimate value if the condition is satisfied and minimum value is the second auxiliary value.

58. (ORIGINAL) A method according to claim 56, wherein the first auxiliary value for the *n*th unknown variable is initially set to be equal to the negative of the right hand side of the *n*th equation.

59. (ORIGINAL) A method according to claim 57, wherein the first auxiliary value for the *n*th unknown variable is initially set to be equal to the negative of the right hand side of the *n*th equation.

60. (ORIGINAL) A method according to claim 55, wherein the respective first and second auxiliary values are updated if the condition is satisfied.

61. (ORIGINAL) A method according to claim 60, wherein the first and second auxiliary values associated with each estimate value are updated if the condition is satisfied.

62. (ORIGINAL) A method according to claim 61, wherein if the predetermined condition is satisfied for the *n*th estimate value:

the first auxiliary value for the *m*th estimate value is updated by:

multiplying the coefficient of the *m*th unknown variable in the *n*th equation by the scalar value *d*; and

adding the result of said multiplication to the first auxiliary value to create a new first estimate auxiliary value, or subtracting the result of said multiplication from the first auxiliary value to create a new first estimate auxiliary value; and

the second auxiliary value for the *m*th estimate value is updated to be equal to the negative of the new first auxiliary value.

63. (ORIGINAL) A method according to claim 39, wherein each estimate value is represented as a fixed point binary word.

64. (ORIGINAL) A method according to claim 39, wherein each estimate value is a floating point binary word.

65. (ORIGINAL) A method according to claim 39, wherein each estimate value is a complex number.

66. (ORIGINAL) A method according to claim 39, wherein step (e) is carried out until a predetermined condition is satisfied.

67. (ORIGINAL) A method according to claim 66, wherein said predetermined condition is a maximum number of iterations without an update to the scalar value *d*.

68. (ORIGINAL) A method according to claim 66, wherein said predetermined condition is a total execution time elapsed without an update to the scalar value *d*.

69. (ORIGINAL) A method according to claim 39, wherein the accurate solution of the equations is known to lie between upper and lower bounds, and the algorithm seeks a solution between said upper and lower bounds.

70. (ORIGINAL) A method according to claim 39, wherein said estimate values are initialised to a value which is with said upper and lower bounds.

71. (ORIGINAL) A method according to claim 70, wherein said estimate values are initialised to a value positioned at the midpoint of said upper and lower bounds.

72. (ORIGINAL) A computer apparatus for solving a system of *N* linear equations in *N* unknown variables, the apparatus comprising:

a program memory containing processor readable instructions; and

a processor for reading and executing the instructions contained in the program memory;

wherein said processor readable instructions comprise instructions controlling the processor to carry out the method according to claim 39.

73. (ORIGINAL) A data carrier carrying computer readable program code to cause a computer to execute procedure in accordance with the method of claim 39.

74. (CANCELLED)

75. (CURRENTLY AMENDED) A computer processor configured to solve a system of N linear equations in N unknown variables, comprising:

storage means for storing an estimate value for each unknown variable;

storage means for storing coefficients of each unknown variable in each equation;

storage means for storing a scalar value d;

initialising means for initialising each estimate value;

computing means configured to process each estimate value by determining whether a respective predetermined condition is satisfied, and to update the estimate if and only if the respective predetermined condition is satisfied, said computing means being configured to repeatedly process each estimate value, and to output said estimate values to provide an estimate of a solution to said system of linear equations ~~until each estimate value is sufficiently close to an accurate value of the respective unknown variable~~.

76. (ORIGINAL) A computer processor according to claim 75, further comprising update means for updating the scalar value d.

77. (ORIGINAL) A computer processor according to claim 76, wherein said update means divides the value of the scalar value d by a value equal to a power of two.

78. (ORIGINAL) A computer processor according to claim 77, wherein said update means divides the value of the scalar value d by a value equal to two.

79. (ORIGINAL) A computer processor according to claim 77, wherein said update means is a bit shift device.

80. (CURRENTLY AMENDED) A computer processor configured to solve a system of N linear equations in N unknown variables, comprising:

storage means for storing an estimate value for each unknown variable;

storage means for storing coefficients of each unknown variable in each equation;

storage means for storing a scalar value *d*;

initialising means for initialising each estimate value;

computing means configured to:

(a) attempt to update each estimate value using a scalar value *d*,

(b) update the scalar value d if no updates are made in step (a); ~~and~~

(c) repeat step (a) and step (b); and

(d) output said estimate values to provide an estimate of a solution to said system of linear equations ~~until each estimate value is sufficiently close to an accurate value of the respective unknown variable~~.

81.-122. (CANCELLED)